SWFT Enhancement

Add support for Quicker digitizer

Thu, Oct 28, 1999

The SWFT local applicationnis used to shepherd waveform measurements using the Swift digitizer that provides digitization rates up to 800KHz. A new variation on the Swift digitizer hardware is the Quicker digitizer that provides digitization rates of up to 20MHz. The SWFT local application can be modified to optionally support this new digitizer. The hardware is accessed via an IndustryPack board interface. A 3-bit code specifies the digitization rate, where 0 specifies the maximum rate, and successively greater values specify factors of two slower than the maximum rate. Here are the possible choices available:

Code	Swift (KHz)	Quicker (KHz)
0	800	20000
1	400	10000
2	200	5000
3	100	2500
4	50	1250
5	25	625
6	12	312
7	6	156

The SWFT local application operates on a data stream queue to accept commands that describe digitization measurements to be performed. It takes each command in turn, drives the digitizer appropriately to make the measurement, waits for the following cycle, then checks the queue for the next measurement The reason for the wait until the next 15Hz cycle is to give a chance for a user of the data to capture it.

The usual client user for this data is FTPM, the local application that supports the Acnet Fast Time Plot interface called FTPMAN. It accepts Acnet requests, and if they are for one of these Swift-style digitizers, passes each into a data stream queue, including the rate to be used, the clock event, the delay, and the number of points to be digitized. FTPM knows about these two digitizers, so it will choose a digitization rate from the above table that is nearest the rate specified by the Acnet user. It will on subsequent cycles check for the completion of the measurement by monitoring fields in the header of the data stream queue. When it sees that the digitization has completed, it captures the data from the hardware buffer and reports completion status to the Acnet client, which is then free to call for the captured data, which remains available until the Acnet request is canceled. The client must capture the data right away, else the hardware may be set up to collect the next waveform requested by an entry in the queue. FTPM simultaneously supports waveform requests—called snapshots by Acnet—for multiple users.

SWFT uses parameters that include the base address of the IP board registers. This address is assumed to be on a 256-byte boundary, so that only the most significant 24 bits are used for the register base address. That means that the low 8 bits may be used for something else. For the Swift digitizer, the low 8 bits are `0x00`. For the Quicker variation, let us assume a value for the low 8 bits of the register base address of `0x10`. (This choice is consistent with one that may be used with the corresponding CINFO table entries to specify the Quicker variation.) The behavior of SWFT is only affected in minor ways by the Quicker choice. Besides the range of possible digitize rates, the monitoring of the address register to determine when the digitization is complete is different. In the case of the Swift digitizer, the address register counts to `0x8000` after storing 4K digitized values for each of the 8 channels. In the case of Quicker, the address register counts to the number of points stored for each channel. If 4K points were digitized, the address register would reach `0x1000`, which is 4K. The maximum memory availablfor the Swift case, per channel, is 4K values. The maximum memory available for the Quicker case per channel, is 16K values. (This implies that the Quicker digitizer total memory is 256K bytes altogether, which allows for 16K words for each of 8 channels.) To capture 16K words of data for each requested channel, and then to deliver 16K words of memory for each channel seems like a heavy load.

`FTPMAN` searches the `CINFO` table to determine what digitization possibilities exist for a specified channel number. The Swift-type entry in `CINFO` specifies the base address of the registers and the channel #0–7 on the board. Using the above suggestion, it can also specify whether the board supports the Swift or the Quicker digitizer. Assume that only one is supported in any one node. This assumption means we can use only a single data stream queue to pass commands to `SWFT`, and to be monitored by `FTPMAN`.

To support a Swift/Quicker digitizer that is not plugged into the CPU board, but is rather on a VME IP carrier board, we must have some way of determining where the memory is that holds the waveform data. Currently, the IP board is assumed to be on the CPU board, so that we can interrogate the IPIC chip to find out where the memory is for each IndustryPack socket on the CPU board. If it is not available through the IPIC, we must provide it by an expansion of the `CINFO` entry. `SwiftMem` is a function that returns the memory address for a given chanel's `CINFO` entry. The memory address for a channel can be found via an optional field in this entry. Each channel uses 4K words for the Swift digitizer, but 16K words for the Quicker digitizer.

Classic protocol approach

To provide a Classic protocol support for Swift digitizer waveforms, a new listype could be defined whose ident would specify the waveform. But when would the data be returned? Would it be useful to return status about the progress of the measurement, which is what `SWFT` provides, after which the client would make a different request to collect the captured data?

Can the data be returned as soon as it can be captured, but not before? This would be a little like event-returned data. There is room for more such definitions in the protocol. If the data is lengthy, how can it be returned in pieces?

How about a variable-length record data stream? A user could monitor records placed there. One record type could give status, but another could include data. The user could write to the `SWFTCMND` data stream the parameters of the measurement. Then monitor what is in the `SWFTSTAT` data stream. If a record is found that indicate status of completion, then the next record will be the data. The completion record could include the info necessary to collect the next record. But to read from a data stream, one must specify a buffer size.